WAF/WOOF Firewall Detection & Analysis Project

Introduction

This project demonstrates how to detect and analyze Web Application Firewalls (WAF) using WAFW00F on Kali Linux within an Oracle VirtualBox environment. Understanding WAF fingerprinting is crucial for security professionals conducting authorized penetration testing and security assessments.

Author: Bolaji Bakare
Date: January 2026
Environment: Kali Linux on Oracle VirtualBox

What is a WAF?

Definition

A **Web Application Firewall (WAF)** is a security solution that monitors, filters, and blocks HTTP/HTTPS traffic to and from web applications. It acts as a shield between web applications and the internet, protecting against common attacks.

How WAFs Work

1. **Traffic Inspection**: Examines all HTTP/HTTPS requests
2. **Rule-Based Filtering**: Applies security rules to detect malicious patterns
3. **Blocking/Allowing**: Blocks suspicious requests, allows legitimate traffic
4. **Logging**: Records all security events for analysis

Common WAF Solutions

- Cloudflare
- AWS WAF
- Akamai Kona
- Imperva (Incapsula)
- F5 BIG-IP ASM

- ModSecurity
- Barracuda WAF
- Fortinet FortiWeb
- Sucuri CloudProxy

What is WAFW00F?

Definition

WAFW00F (Web Application Firewall Fingerprinting Tool) is a Python-based tool that identifies and fingerprints Web Application Firewall products protecting websites. It sends specially crafted HTTP requests and analyzes responses to determine the WAF vendor and version.

Key Features

- Detects over 150+ WAF solutions
- Fast and accurate fingerprinting
- Multiple detection techniques
- Easy-to-use command-line interface
- Supports custom headers and payloads

Project Objectives

1. Install and configure WAFW00F on Kali Linux
2. Learn WAF detection techniques
3. Identify WAF solutions protecting target websites
4. Integrate SpiderFoot for comprehensive reconnaissance
5. Analyze scan results for security assessment
6. Understand security implications and best practices

Prerequisites

System Requirements

- **OS**: Kali Linux (2023.1 or later)
- **Virtualization**: Oracle VirtualBox 6.0+
- **RAM**: Minimum 2GB (4GB recommended)
- **Disk Space**: 20GB free space
- **Network**: Active internet connection

Required Knowledge

- Basic Linux command-line skills
- Understanding of HTTP/HTTPS protocols
- Basic networking concepts
- Familiarity with web security fundamentals

Tools Checklist

- [ ] Kali Linux installed on VirtualBox
- [ ] Python 3.8+ installed
- [ ] pip (Python package manager)
- [ ] Git installed
- [ ] Terminal access

Guide

Step 1: Update Kali Linux System

Update package lists

sudo apt update

Upgrade installed packages

sudo apt upgrade -y

Install system dependencies

sudo apt install -y git python3 python3-pip

What to look for:

- No error messages during update
- All packages successfully upgraded
- Python 3.8+ installed (check with python3 --version)

Step 2: Install WAFW00F

Method 1: Using apt (Recommended)

Install WAFW00F from Kali repositories

sudo apt install wafw00f -y

Verify installation

wafw00f --version

Method 2: Using pip

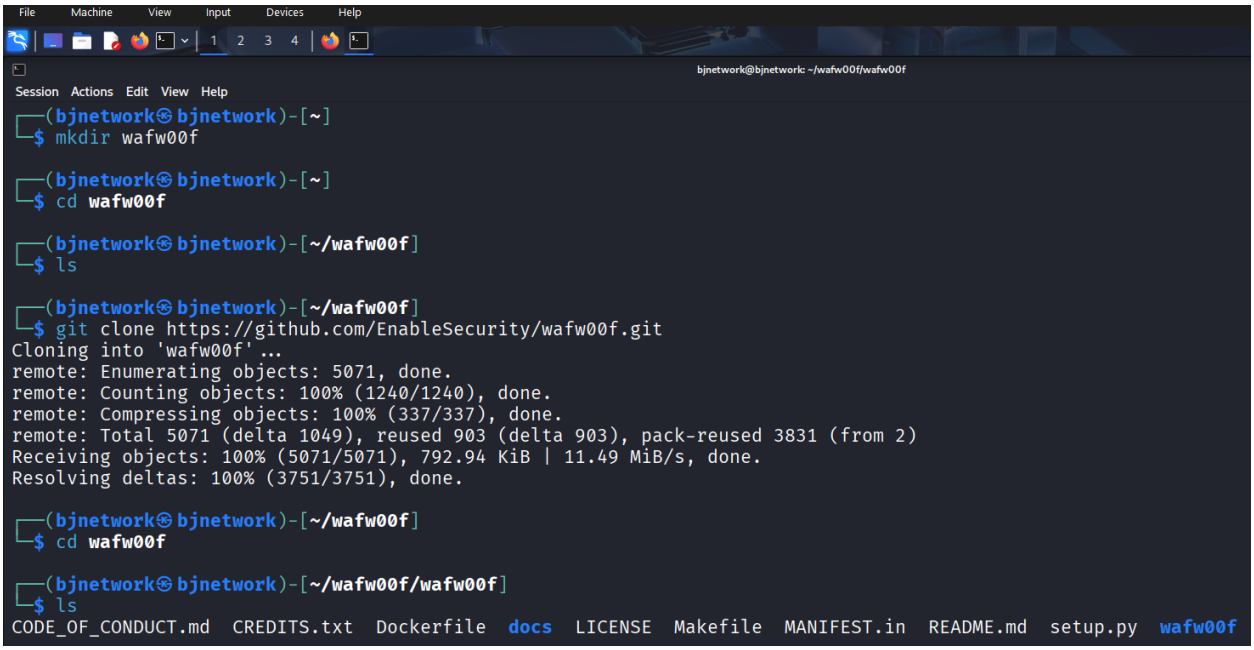
Install using Python pip

sudo pip3 install wafw00f

Verify installation

wafw00f --version

Method 3: From GitHub (Latest Development Version)



Clone the repository

cd ~/Documents

git clone https://github.com/EnableSecurity/wafw00f.git

Navigate to directory

cd wafw00f

Install dependencies

sudo pip3 install -r requirements.txt

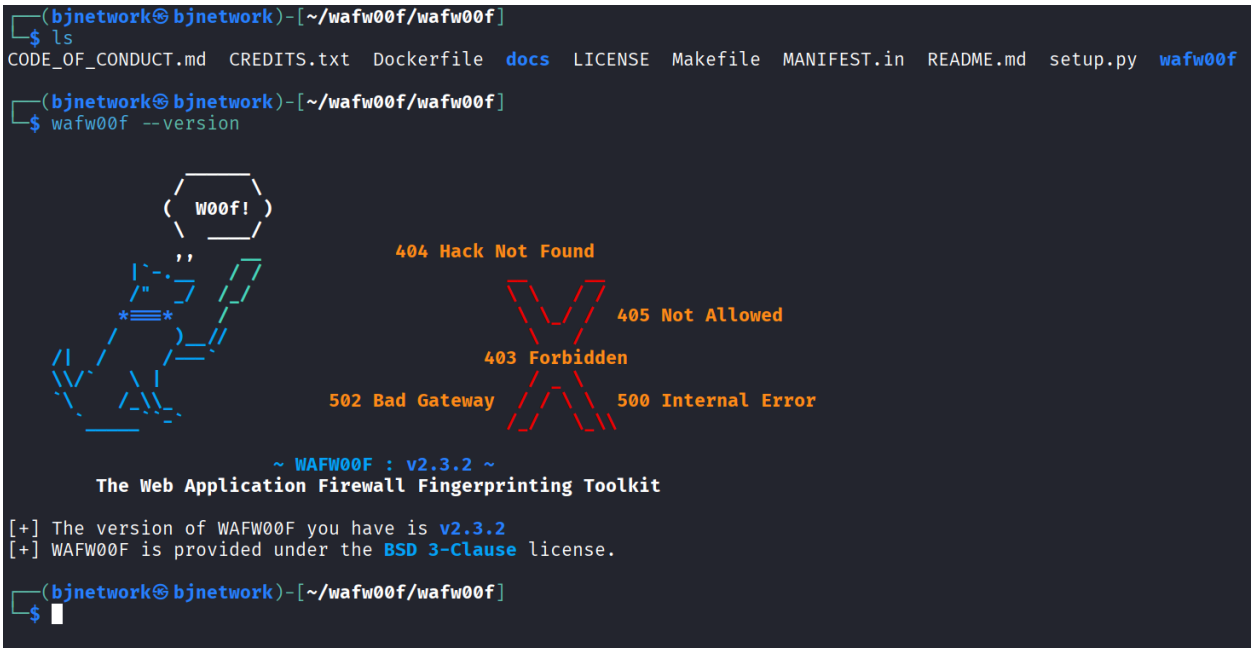
Install WAFW00F

sudo python3 setup.py install

Verify installation

wafw00f --version

Expected Output:



WAFW00F - Web Application Firewall Detection Tool

Version: 2.x.x

Step 3: Install SpiderFoot

Navigate to home directory

cd ~/Documents

Clone SpiderFoot repository

git clone https://github.com/smicallef/spiderfoot.git

Navigate to SpiderFoot directory

cd spiderfoot

Install Python dependencies

sudo pip3 install -r requirements.txt

Verify installation:

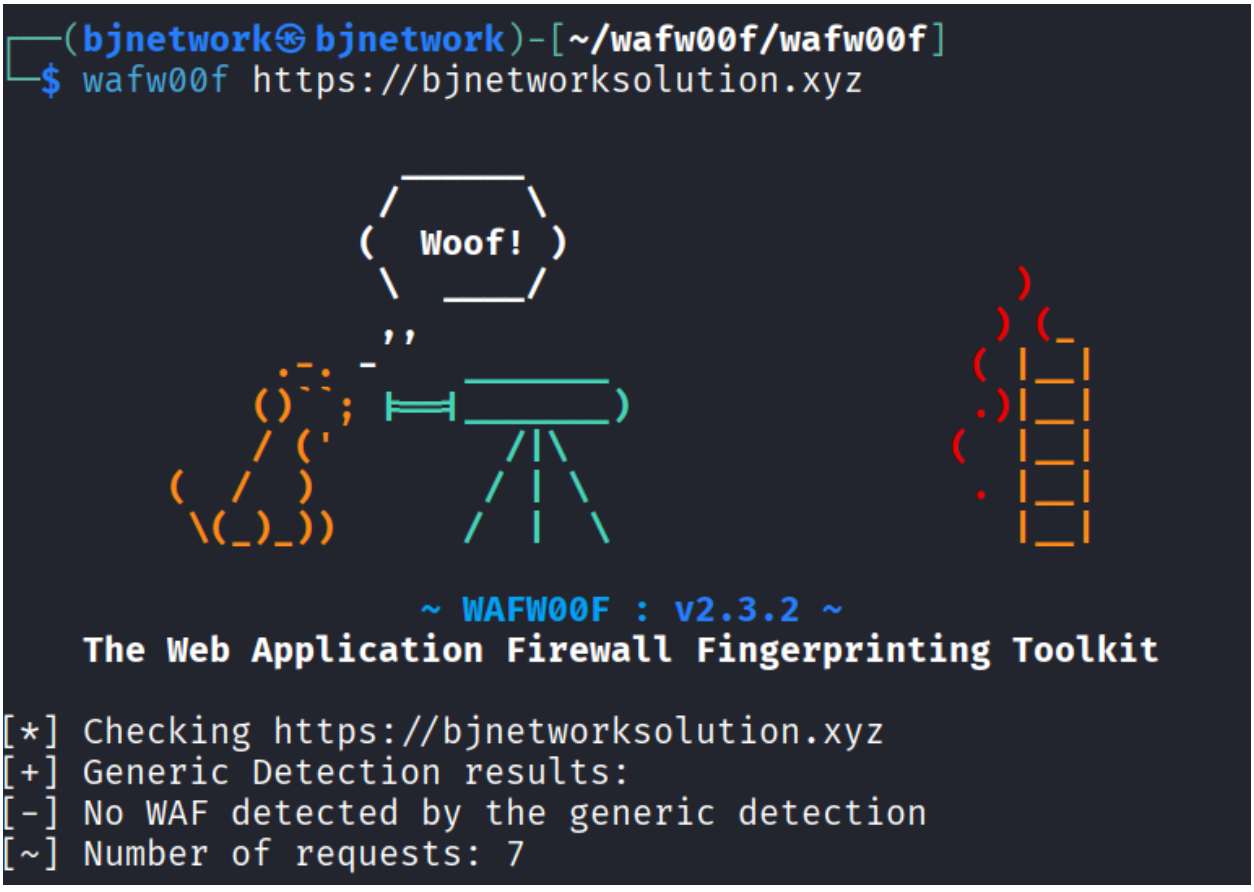
Check SpiderFoot version

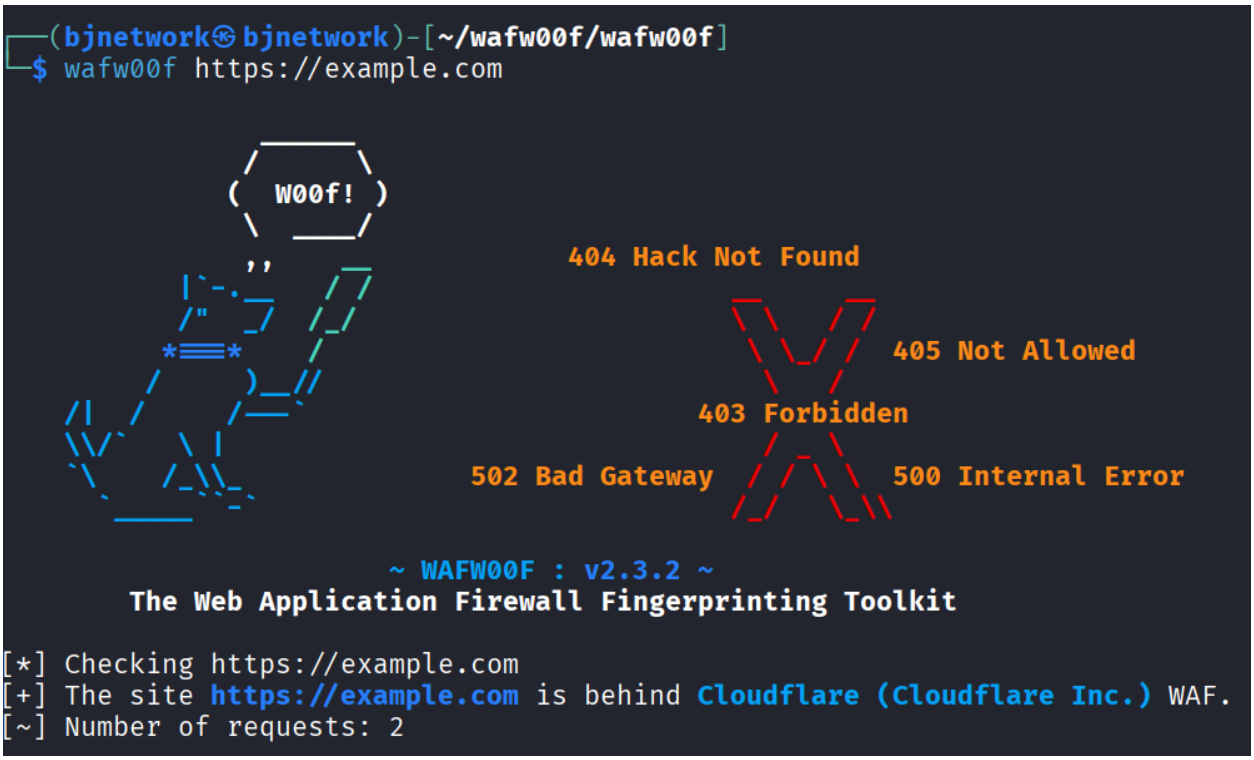
python3 sf.py --version

Step-by-Step Usage Guide

Phase 1: Basic WAF Detection

Step 1: Simple WAF Detection on Single Target





Basic syntax

wafw00f <target_url>

Example

wafw00f https://example.com

What to look for in output:

- WAF detected or not detected
- WAF vendor name (e.g., Cloudflare, AWS WAF)
- Confidence level of detection

- Response time

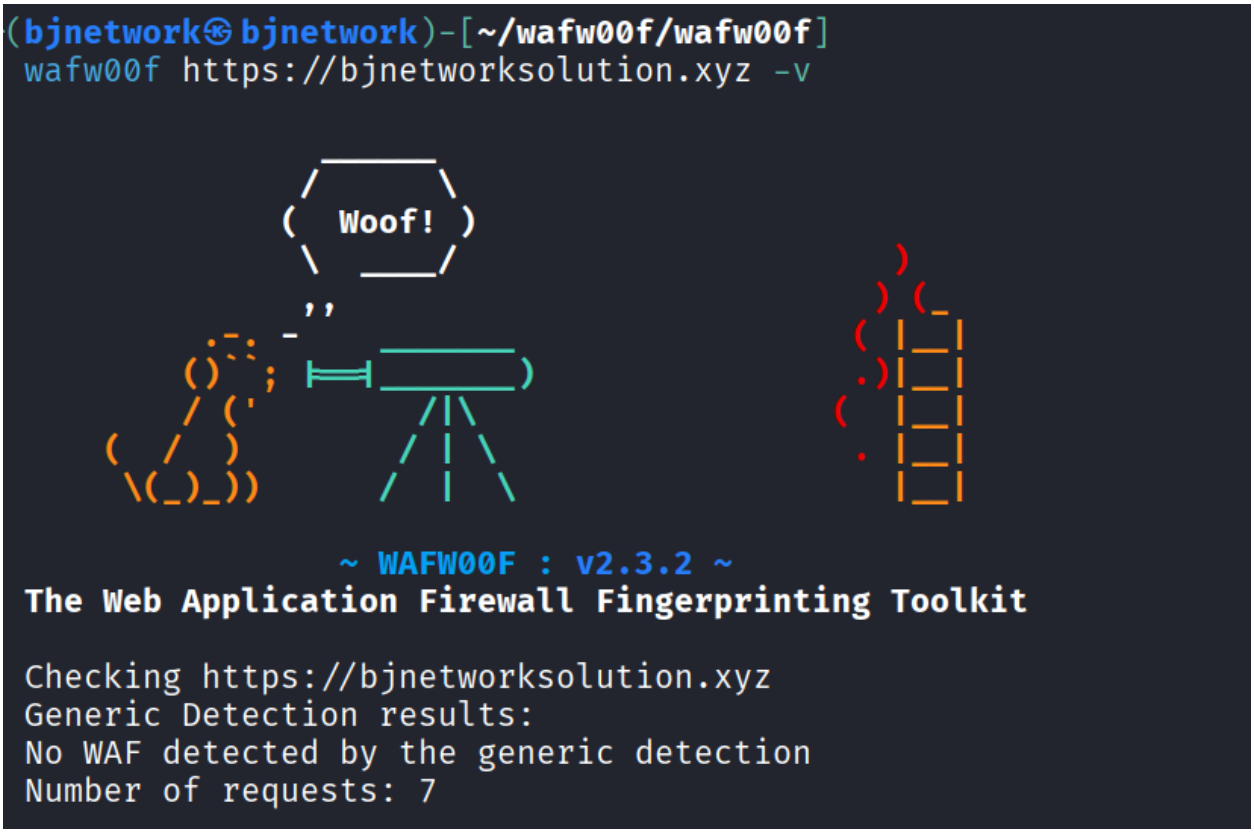
Sample Output Interpretation:

[*] Checking https://example.com

[+] The site https://example.com is behind Cloudflare (Cloudflare Inc.) WAF.

[~] Number of requests: 5

Step 2: Verbose Mode for Detailed Information

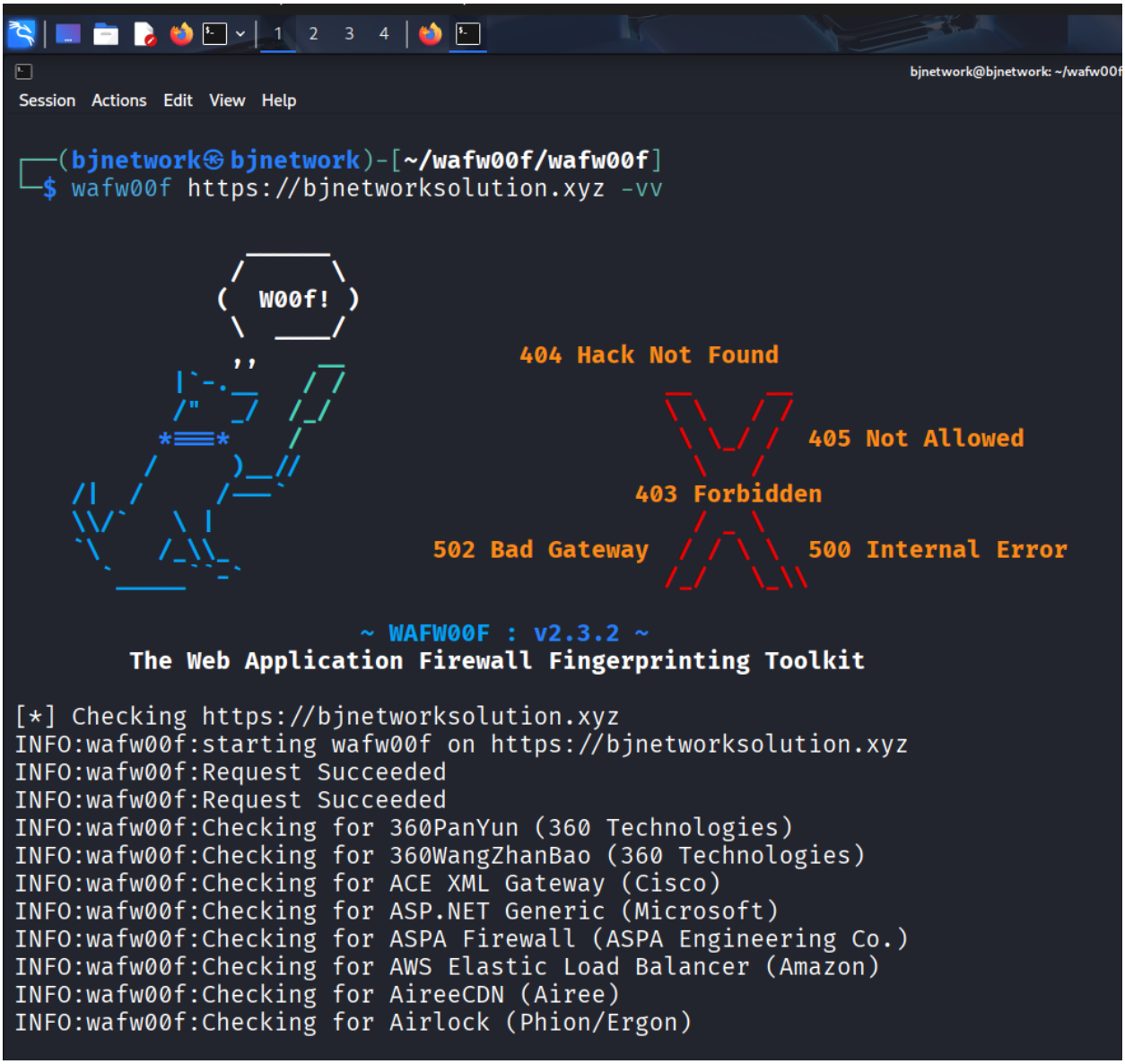


Enable verbose output

wafw00f https://example.com -v

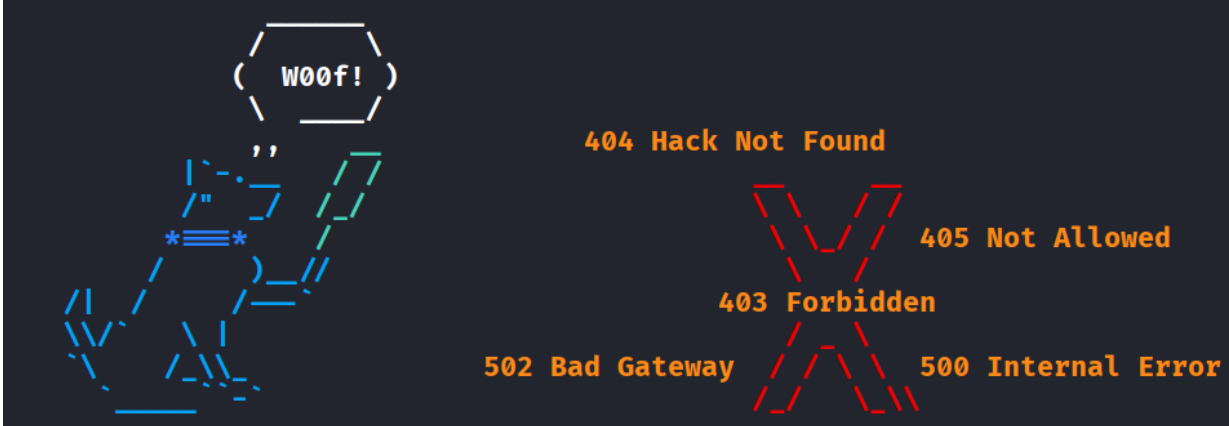
Maximum verbosity

wafw00f https://example.com -vv

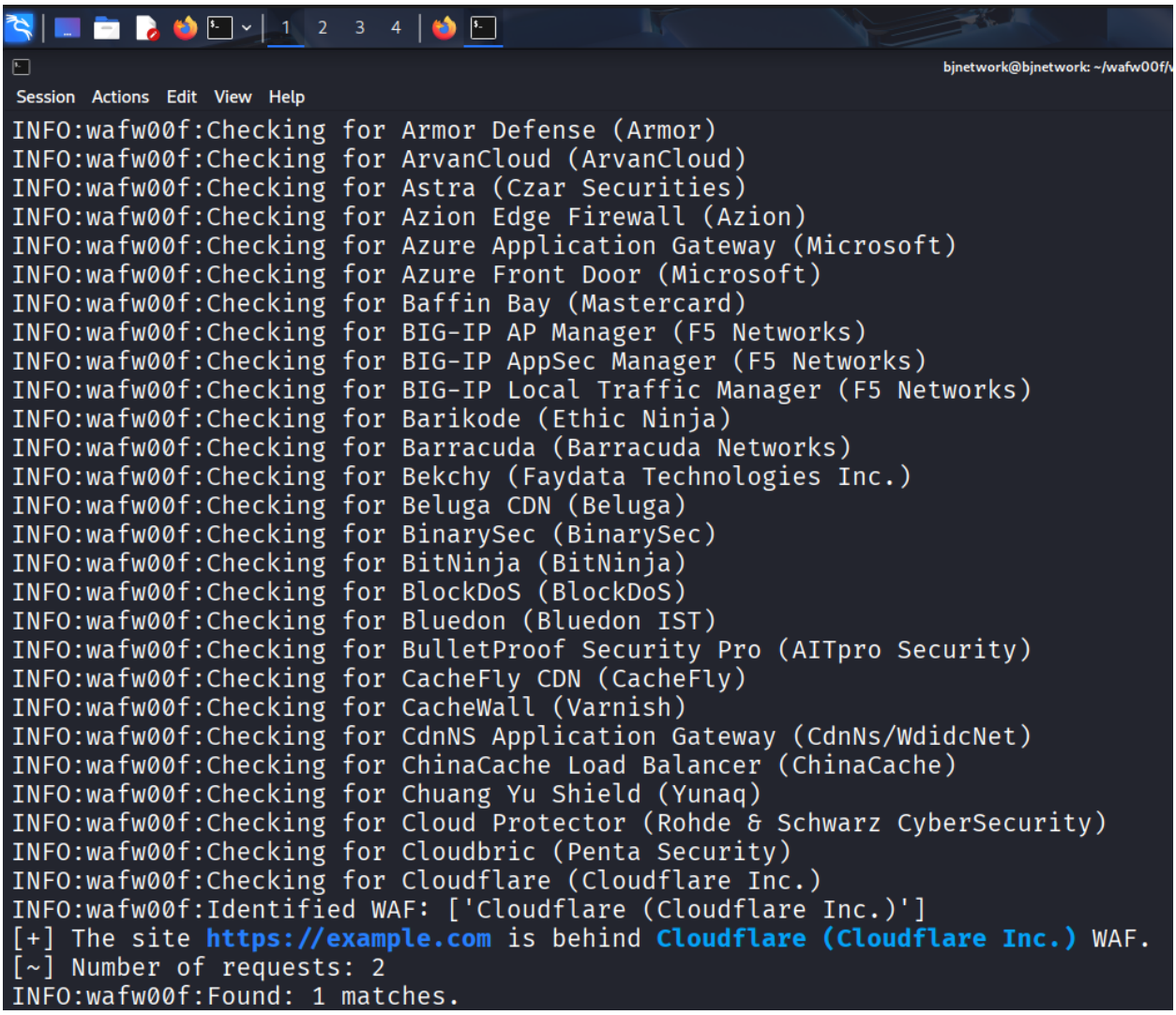


```
INFO:wafw00f:Checking for WebARX (WebARX Security Solutions
INFO:wafw00f:Checking for WebKnight (AQTRONIX)
INFO:wafw00f:Checking for WebLand (WebLand)
INFO:wafw00f:Checking for WebSEAL (IBM)
INFO:wafw00f:Checking for WebTotem (WebTotem)
INFO:wafw00f:Checking for West263 CDN (West263CDN)
INFO:wafw00f:Checking for Wordfence (Defiant)
INFO:wafw00f:Checking for XLabs Security WAF (XLabs)
INFO:wafw00f:Checking for Xuanwudun (Xuanwudun)
INFO:wafw00f:Checking for YXLink (YxLink Technologies)
INFO:wafw00f:Checking for Yundun (Yundun)
INFO:wafw00f:Checking for Yunjiasu (Baidu Cloud Computing)
INFO:wafw00f:Checking for Yunsuo (Yunsuo)
INFO:wafw00f:Checking for ZScaler (Accenture)
INFO:wafw00f:Checking for Zenedge (Zenedge)
INFO:wafw00f:Checking for aeSecure (aeSecure)
INFO:wafw00f:Checking for eEye SecureIIS (BeyondTrust)
INFO:wafw00f:Checking for pkSecurity IDS (pkSec)
INFO:wafw00f:Checking for wpmudev WAF (Incsub)
INFO:wafw00f:Checking for Shieldon Firewall (Shieldon.io)
INFO:wafw00f:Identified WAF: []
[+] Generic Detection results:
INFO:wafw00f:Request Succeeded
INFO:wafw00f:Request Succeeded
INFO:wafw00f:Request Succeeded
INFO:wafw00f:Request Succeeded
INFO:wafw00f:Request Succeeded
[-] No WAF detected by the generic detection
[~] Number of requests: 7
INFO:wafw00f:Found: 1 matches.
```

```
┌──(bjnetwork㉿bjnetwork)-[~/wafw00f/wafw00f]
└─$ wafw00f https://example.com -vv
```



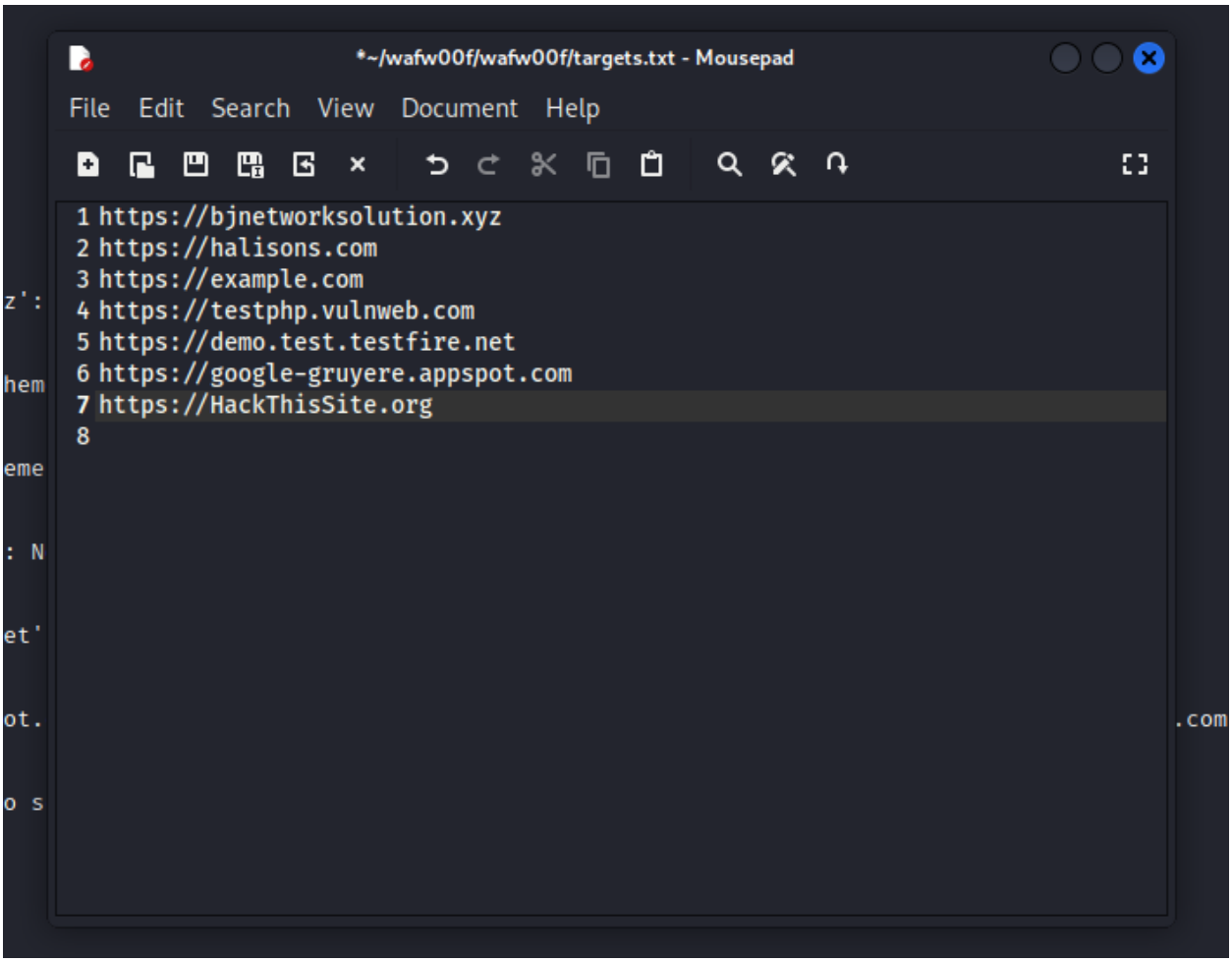

```
                    ~ WAFW00F : v2.3.2 ~
        The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://example.com
[INFO:wafw00f:starting wafw00f on https://example.com
[INFO:wafw00f:Request Succeeded
[INFO:wafw00f:Request Succeeded
[INFO:wafw00f:Checking for 360PanYun (360 Technologies)
[INFO:wafw00f:Checking for 360WangZhanBao (360 Technologies)
[INFO:wafw00f:Checking for ACE XML Gateway (Cisco)
[INFO:wafw00f:Checking for ASP.NET Generic (Microsoft)
[INFO:wafw00f:Checking for ASPA Firewall (ASPA Engineering Co.)
[INFO:wafw00f:Checking for AWS Elastic Load Balancer (Amazon)
[INFO:wafw00f:Checking for AireeCDN (Airee)
```



Key information revealed:

- HTTP headers sent and received
- Detection techniques used
- Response codes
- Server fingerprints

Step 3: Testing Multiple Targets



Create a targets file:

Create targets.txt

nano targets.txt

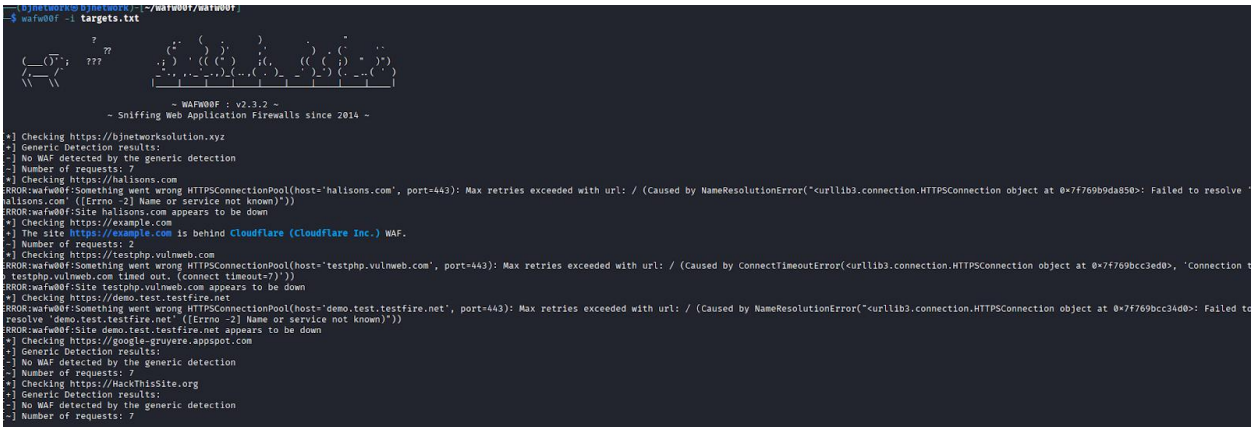
Add targets (one per line):

https://example1.com

https://example2.com

https://example3.com

Run scan:



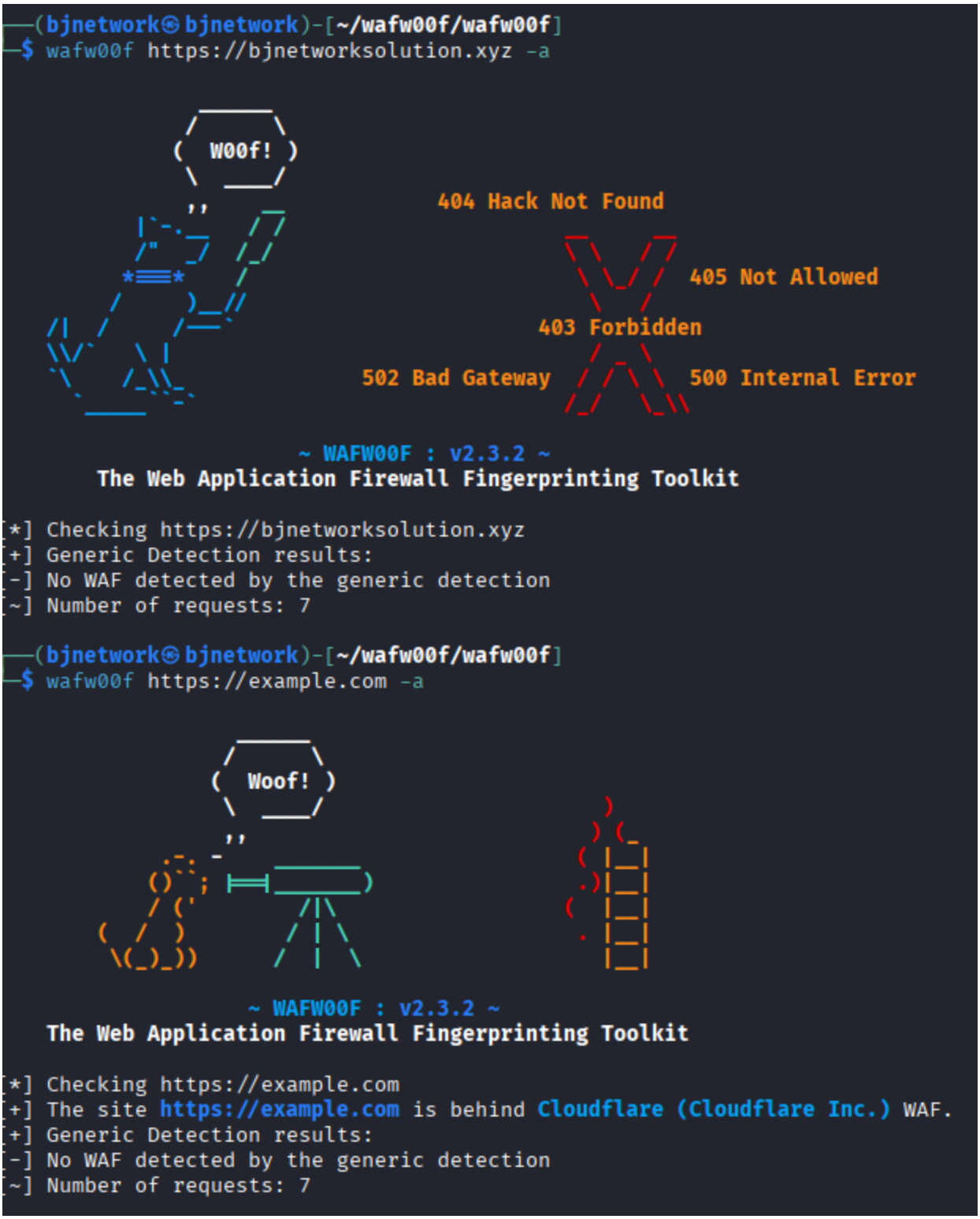
Scan multiple targets

wafw00f -i targets.txt

With output file

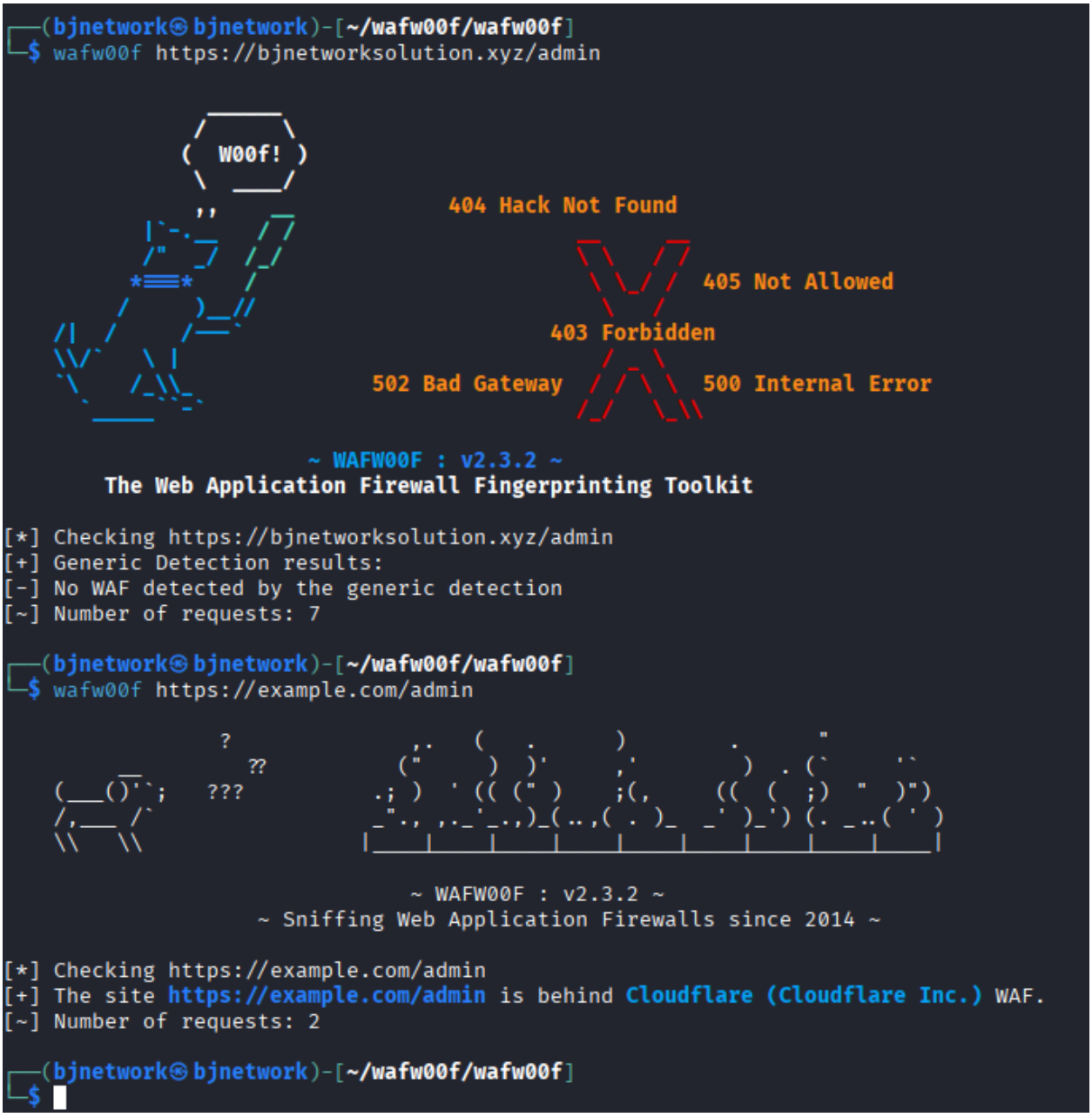
wafw00f -i targets.txt -o results.txt

Step 4: Advanced Detection Techniques



Test all detections (more thorough)

wafw00f https://example.com -a



Check specific path

wafw00f https://example.com/admin

Use custom User-Agent

wafw00f https://example.com -H "User-Agent: CustomBot/1.0"

Specify proxy

wafw00f https://example.com -p http://127.0.0.1:8080

Step 5: Output Formats

Save to text file

wafw00f https://example.com -o scan_results.txt

JSON format (for automation)

wafw00f https://example.com -f json -o results.json

CSV format

wafw00f https://example.com -f csv -o results.csv

Phase 2: Comprehensive Scanning with Different Options

All Available Commands Reference

Display help

wafw00f -h

List all supported WAFs

wafw00f -l

Version information

wafw00f --version

Test specific WAF (if you suspect one)

wafw00f https://example.com -t "Cloudflare"

Disable SSL certificate verification (use cautiously)

wafw00f https://example.com --no-ssl-verify

Set custom timeout (in seconds)

wafw00f https://example.com --timeout 10

Follow redirects

wafw00f https://example.com --follow-redirects

Phase 3: Analyzing Scan Results

What Each Result Means

1. WAF Detected:

[+] The site https://example.com is behind Cloudflare WAF.

Interpretation:

- Site is protected by a WAF
- Attacks may be filtered
- Adjust testing methodology accordingly

2. No WAF Detected:

[-] The site https://example.com does not seem to be behind a WAF.

Interpretation:

- No WAF identified (or WAF is well-hidden)
- Direct application testing possible
- Still check for other security measures

3. Generic Detection:

[+] Generic Detection: 80%

Interpretation:

- WAF likely present but vendor unknown
- May be custom WAF solution
- Requires manual investigation

Important Information to Look For

During Scanning

1. HTTP Response Headers

Look for these WAF indicators:

Use curl to inspect headers

curl -I https://example.com

Key headers:

- Server: - May reveal WAF (e.g., "cloudflare")
- X-CDN: - CDN/WAF identification
- X-Powered-By: - Technology stack
- Set-Cookie: - WAF session cookies (e.g., "__cfduid")

2. Response Codes

- 403 Forbidden - Possible WAF block
- 406 Not Acceptable - WAF rejection
- 503 Service Unavailable - Rate limiting or blocking
- 200 OK with unusual content - WAF challenge page

3. Response Time Patterns

Measure response times

time wafw00f https://example.com

- Consistent delays may indicate WAF processing
- Timeouts suggest aggressive filtering

After Scanning

1. WAF Presence Indicators

Strong indicators:

- Specific WAF signature detected
- Proprietary HTTP headers
- Known WAF cookie names
- Characteristic error pages

Weak indicators:

- Generic security headers
- Cloud hosting provider
- Fast response times

2. Technology Stack Information

Document findings

nano technology_stack.txt

Record:

- Web server type and version
- Programming languages
- Frameworks detected
- Third-party services (CDN, analytics)
- CMS platforms

3. Security Headers Analysis

Check for security headers:

- Strict-Transport-Security (HSTS)
- Content-Security-Policy (CSP)
- X-Frame-Options

- X-Content-Type-Options
- X-XSS-Protection

4. SSL/TLS Configuration

Use SSL Labs for detailed analysis

Or use testssl.sh

cd ~/Documents

git clone https://github.com/drwetter/testssl.sh.git

cd testssl.sh

./testssl.sh https://example.com

Practical Applications

1. Penetration Testing & Security Assessments

Use Case: Understanding WAF presence helps pentesters adjust their testing methodology.

How to use the information:

Identify WAF

wafw00f https://client-site.com -o waf_report.txt

Adjust testing tools based on WAF

Example: Use slower, more evasive scans

nmap -sS -T2 --script http-waf-detect target_ip

Best Practices:

- Always get written authorization
- Document WAF detection in reports
- Recommend WAF bypasses to clients (for remediation)

- Test WAF effectiveness with authorized payloads

2. Red Team Operations

Use Case: Reconnaissance phase to understand defensive capabilities.

Information utilization:

1. **Plan attack vectors:**
    - If no WAF detected: Standard exploitation techniques
    - If WAF detected: Evasion techniques required
2. **Select tools accordingly:**
    - WAF-aware scanners (e.g., sqlmap with tamper scripts)
    - Custom payloads for specific WAFs
    - Timing and rate limiting considerations

Command example:

Use sqlmap with WAF detection

```
sqlmap -u "https://example.com/page?id=1" --identify-waf --random-agent
```

3. Bug Bounty Hunting

Use Case: Efficient target reconnaissance for vulnerability research.

Workflow:

Step 1: Detect WAF

```
wafw00f https://bugbounty-target.com -v
```

Step 2: If WAF detected, research known bypasses

Document in notes

Step 3: Focus on business logic flaws (often bypass WAF)

Strategy:

- WAF presence = focus on logic flaws, authentication issues
- No WAF = broader vulnerability testing possible
- Document WAF config issues as findings

4. Security Monitoring & Blue Team

Use Case: Verify WAF deployment and configuration.

How to use:

Periodically scan your own infrastructure

wafw00f https://your-company.com -o monthly_check.txt

Compare results over time

diff last_month_waf.txt this_month_waf.txt

Monitoring checklist:

- [ ] WAF still active and detected
- [ ] WAF signatures up to date
- [ ] No unexpected changes in configuration
- [ ] Response headers consistent

5. Compliance & Audit Purposes

Use Case: Verify security controls for compliance (PCI-DSS, HIPAA, etc.).

Documentation:

Generate compliance report

wafw00f https://compliance-target.com -f json -o compliance_report.json

Include in audit documentation

Note: WAF presence addresses specific compliance requirements

Advantages & Disadvantages

Advantages of WAF Detection

For Security Professionals

1. Informed Testing Methodology

- Adjust penetration testing approaches
- Select appropriate tools and techniques
- Avoid triggering unnecessary blocks

2. Efficient Reconnaissance

- Quick identification of defensive layers
- Understanding of security posture
- Resource allocation for testing

3. Better Reporting

- Complete security assessment
- Accurate risk evaluation
- Comprehensive recommendations

4. Compliance Verification

- Confirm required controls
- Audit security implementations
- Meet regulatory requirements

Advantages of Using WAFs

1. Protection Against Common Attacks

- SQL injection blocking
- Cross-Site Scripting (XSS) prevention
- CSRF protection
- DDoS mitigation

2. Centralized Security Management

- Single point for security rules
- Easy policy updates
- Consistent protection across applications

3. Virtual Patching

- Protect unpatched vulnerabilities
- Buy time for proper remediation
- Reduce attack surface

4. Compliance Support

- Meet regulatory requirements (PCI-DSS 6.6)
- Logging and monitoring capabilities
- Audit trail generation

5. Bot Protection

- Block malicious bots
- Prevent scraping
- Rate limiting

Disadvantages of WAF Detection Tools

1. False Positives/Negatives

- May not detect all WAF types
- Custom WAFs might be missed
- Some WAFs designed to be stealthy

2. Limited Information

- Only identifies presence, not configuration
- Doesn't reveal WAF rules
- No insight into bypass techniques

3. Detection Can Be Blocked

- Some WAFs detect fingerprinting attempts
- May trigger security alerts
- IP blocking risk

4. Requires Updates

- New WAFs released regularly
- Signature database needs maintenance
- May miss latest WAF versions

Disadvantages of WAFs

1. Performance Impact

- Added latency to requests
- Resource consumption
- Potential bottleneck

2. False Positives

- Legitimate traffic blocked
- User frustration
- Business impact

3. Maintenance Overhead

- Rule tuning required
- Regular updates needed
- Expertise required for optimization

4. Not a Complete Solution

- Doesn't fix underlying vulnerabilities
- Can be bypassed
- Shouldn't replace secure coding

5. Cost

- Commercial WAFs expensive
- Operational costs (personnel, time)
- Infrastructure requirements

Security Best Practices

For Using WAFW00F & SpiderFoot

1. Legal Authorization

ALWAYS create authorization documentation

Template:

nano authorization_letter.txt

Required elements:

- Written permission from target owner
- Scope definition (URLs, IP ranges)
- Testing timeframe
- Contact information
- Signature and date

Never scan without permission - it's illegal!

2. Rate Limiting & Respectful Scanning

Use delays between requests

wafw00f https://example.com --timeout 5

For multiple targets, add delays

```
for target in $(cat targets.txt); do
    wafw00f $target
    sleep 5  # Wait 5 seconds between scans
done
```